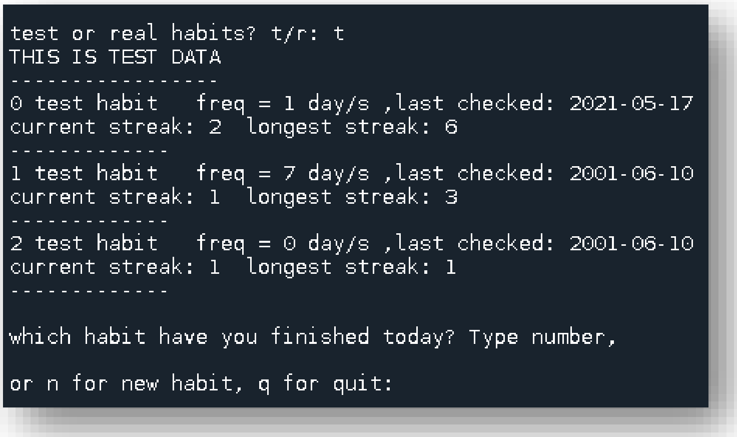
Habit tracking app

Development/reflection phase

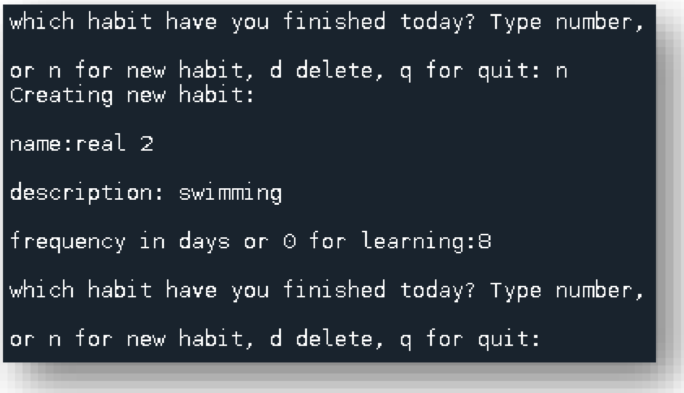
How it works-basics

Program is named HabitTracker.py and can be downloaded from https://github.com/vldasx/Habit-tracking-app. It can be loaded and run in some Python Integrated development environment such as Spyder or PyCharm.

This is a small habit tracker with a very simple user interface, and communicates with user through text UI. All data is organized in one list of objects (habits) and the first 3 of them are test data sets, 4th is "real" data set. User first chooses the test or real set. Analytical tool calculates current and longest streak at the beginning, and also later when data changes. It does that for a selected set of data. It is shown as a motivator and stimulation to accomplish daily tasks (pic. 1). The program expects an input, and we can enter a number that represents a habit. This will automatically save today's date in this habit. We can repeat this process as many times as we like, even multiple times for the same habit. Program assumes that the user has accomplished a habit more than once, it will save it and ignore duplicates during calculations. There are three more options for input, n as new habit, d delete and q for quit. Quitting the program will erase all data, and this is exit meant for program testing. When typing n, that is going to create a new habit data set (pic. 2). Typing d will give a menu for deleting (real) habits. When all is finished, it goes again to the main menu.



```
test or real habits? t/r: t
THIS IS TEST DATA
-----------------
0 test habit   freq = 1 day/s ,last checked: 2021-05-17
current streak: 2  longest streak: 6
-------------
1 test habit   freq = 7 day/s ,last checked: 2001-06-10
current streak: 1  longest streak: 3
-------------
2 test habit   freq = 0 day/s ,last checked: 2001-06-10
current streak: 1  longest streak: 1
-------------

which habit have you finished today? Type number,

or n for new habit, q for quit:
```

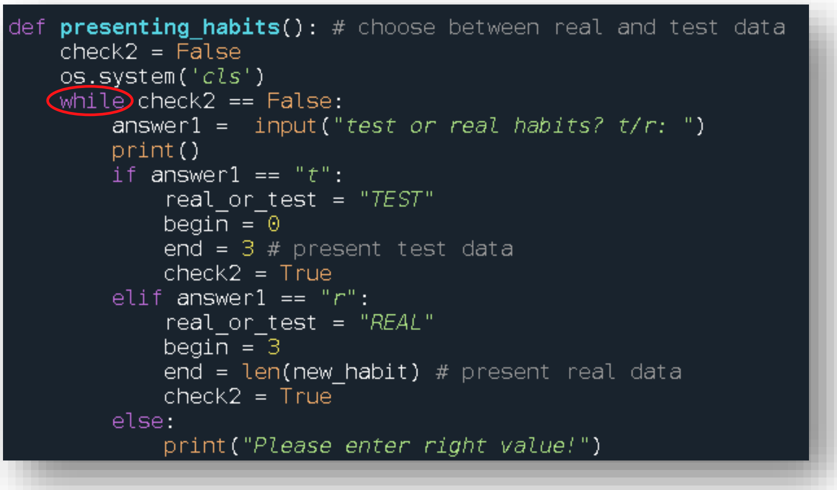


```
which habit have you finished today? Type number,

or n for new habit, d delete, q for quit: n
Creating new habit:

name:real 2

description: swimming

frequency in days or 0 for learning:8

which habit have you finished today? Type number,

or n for new habit, d delete, q for quit:
```

Some technical details:

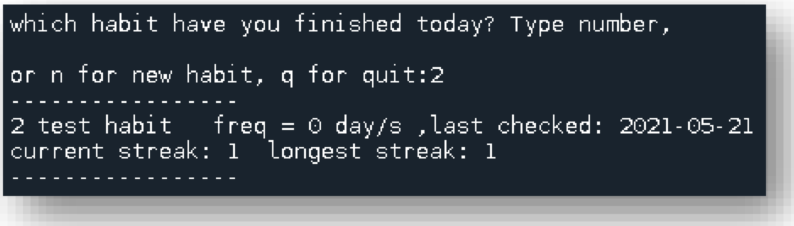
1. Tracking of one habit begins on a first inserted/saved date. Period for tracking is firm, which means that once started, it is expected to insert a date that is between the end of previous and end of the next cycle. This adds one to the current streak. If the date is smaller than that, the date is saved but it will be ignored during analysis. If the date is bigger, the streak is broken and a new count for the current streak begins.

2. Constraints- while loop (pic. 3) is used to check if the first input is valid. If it is not valid, the user is asked to enter the right value and the loop restarts with input, ignoring the wrong one. This method can be used in the program at the places of input, and it is omitted only to simplify the program.



```
def presenting_habits(): # choose between real and test data
    check2 = False
    os.system('cls')
    while check2 == False:
        answer1 =  input("test or real habits? t/r: ")
        print()
        if answer1 == "t":
            real_or_test = "TEST"
            begin = 0
            end = 3 # present test data
            check2 = True
        elif answer1 == "r":
            real_or_test = "REAL"
            begin = 3
            end = len(new_habit) # present real data
            check2 = True
        else:
            print("Please enter right value!")
```

3. Analytics is calculated separately and presented in the main screen as reminder (pic. 1). For every new recorded data analytics for this habit is called again, so the change can be immediately seen, and analytics is not separately shown (pic. 4).

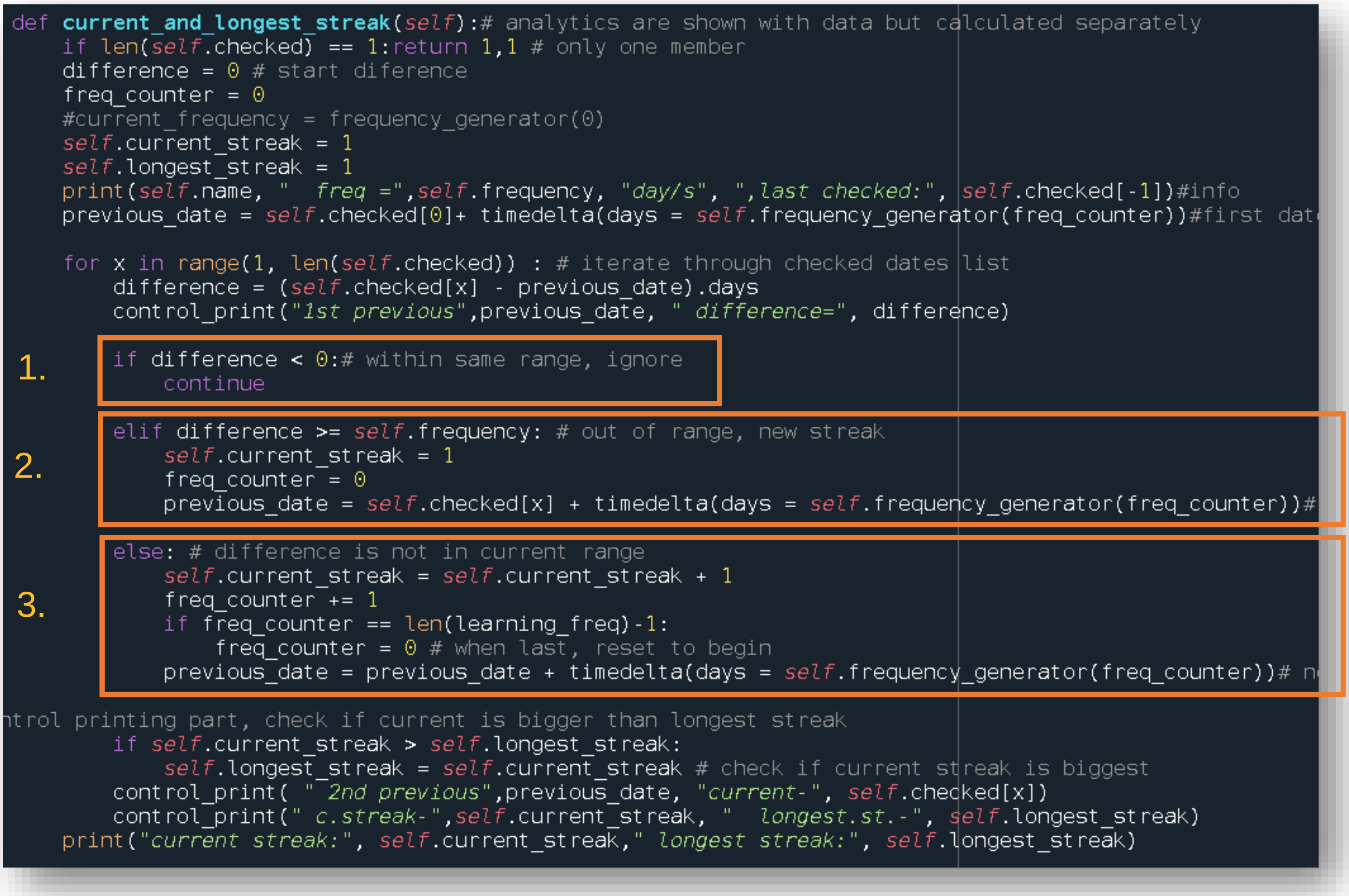


```
which habit have you finished today? Type number,

or n for new habit, q for quit:2
-----------------
2 test habit   freq = 0 day/s ,last checked: 2021-05-21
current streak: 1  longest streak: 1
-----------------
```

4. Variable repeating frequency (learning)- when frequency is set to 0, data for actual frequency is taken from list learning_freq. Those frequencies can be changed according to need, or there can be added some other as choice.

5. Regular repeating frequency can be any whole positive number.

Program:



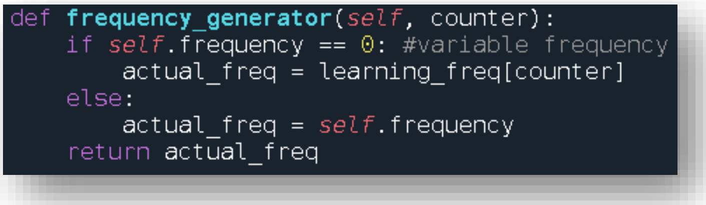
```
def current_and_longest_streak(self):# analytics are shown with data but calculated separately
    if len(self.checked) == 1:return 1,1 # only one member
    difference = 0 # start diference
    freq_counter = 0
    #current_frequency = frequency_generator(0)
    self.current_streak = 1
    self.longest_streak = 1
    print(self.name, "  freq =",self.frequency, "day/s", ",last checked:", self.checked[-1])#info
    previous_date = self.checked[0]+ timedelta(days = self.frequency_generator(freq_counter))#first dat

    for x in range(1, len(self.checked)) : # iterate through checked dates list
        difference = (self.checked[x] - previous_date).days
        control_print("1st previous",previous_date, " difference=", difference)

        if difference < 0:# within same range, ignore
            continue

        elif difference >= self.frequency: # out of range, new streak
            self.current_streak = 1
            freq_counter = 0
            previous_date = self.checked[x] + timedelta(days = self.frequency_generator(freq_counter))#

        else: # difference is not in current range
            self.current_streak = self.current_streak + 1
            freq_counter += 1
            if freq_counter == len(learning_freq)-1:
                freq_counter = 0 # when last, reset to begin
            previous_date = previous_date + timedelta(days = self.frequency_generator(freq_counter))# n
ntrol printing part, check if current is bigger than longest streak
        if self.current_streak > self.longest_streak:
            self.longest_streak = self.current_streak # check if current streak is biggest
        control_print( " 2nd previous",previous_date, "current-", self.checked[x])
        control_print(" c.streak-",self.current_streak, "  longest.st.-", self.longest_streak)
    print("current streak:", self.current_streak," longest streak:", self.longest_streak)
```

Heart of the analytical module is the current_and_longest_streak function encapsulated within the habit object. When called, it iterates through a list of dates of the object and calculates current and longest streaks. Variable previous_date is defined as current date + current frequency, and it serves as boundary, beginning of new period. It is the smallest date in the new set, and that is why it is called so. There are 3 possibilities:

1. Date is smaller than the previous. Meaning that date is in the same range and it is ignored.

2. Date is bigger or equal than the first boundary + current frequency. It breaks the streak and the counter for the streak is reduced to 1 (it is the first member of the new streak), freq_counter is 0 (it is for variable counters, their streak begins with the 0th member in the list).

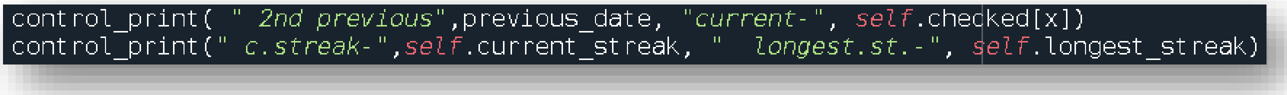
3. When two possibilities are eliminated (date is not smaller or bigger than boundaries) then date is within boundaries, 1 is added to streak, freq_counter is again reduced to 0 if it was the last member of list or stays increased for 1. New previous_date is set.



```python
def frequency_generator(self, counter):
    if self.frequency == 0: #variable frequency
        actual_freq = learning_freq[counter]
    else:
        actual_freq = self.frequency
    return actual_freq
```

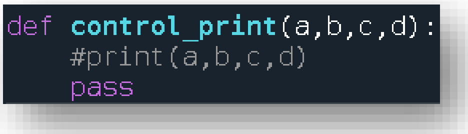
At all places where frequency is used, small function frequency_generator (pic. 5) is called to return a value from the list if it is a variable frequency, or just the plane frequency of that object if it is a fixed period.

At the bottom of the function current_and_longest_streak is the control printing part (pic. 6)



```python
control_print( " 2nd previous",previous_date, "current-", self.checked[x])
control_print(" c.streak-",self.current_streak, "  longest.st.-", self.longest_streak)
```

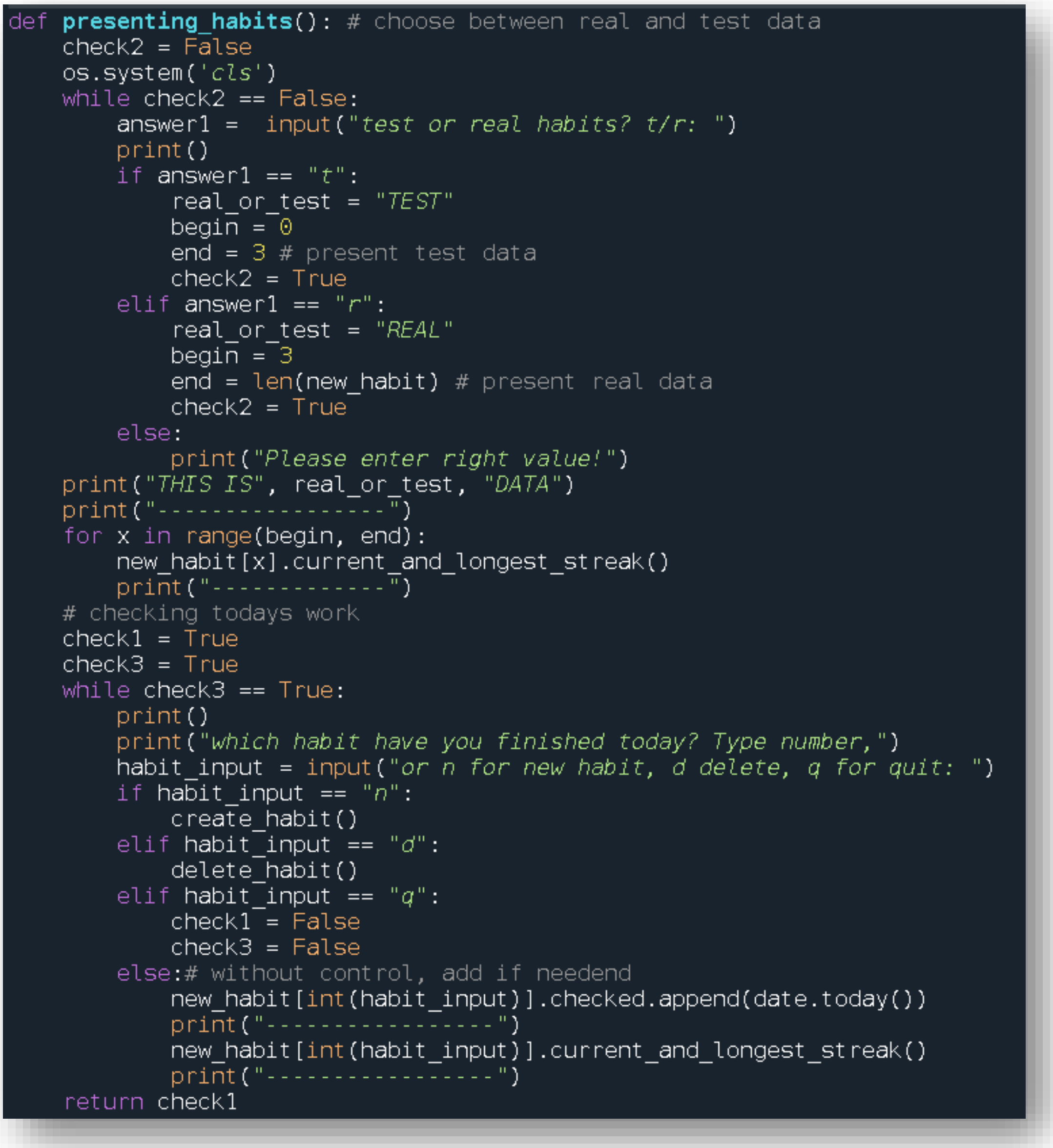
which calls the control_print function (pic. 7). This function serves only as control during testing, it is called from several places in the program and can easily be switched off like here.



```python
def control_print(a,b,c,d):
    #print(a,b,c,d)
    pass
```

Function create_habit is straightforward, takes input and casts a new element. Second function for habit management is delete_habit, which only protects test data from erasing. As all data stays in a list, an erased set will not leave a hole, the list simply moves all data from above, and the last index is free again. It is used when iterating through data with for loop, upper bound is length of this list.

presenting_habits function



```
def presenting_habits(): # choose between real and test data
    check2 = False
    os.system('cls')
    while check2 == False:
        answer1 =  input("test or real habits? t/r: ")
        print()
        if answer1 == "t":
            real_or_test = "TEST"
            begin = 0
            end = 3 # present test data
            check2 = True
        elif answer1 == "r":
            real_or_test = "REAL"
            begin = 3
            end = len(new_habit) # present real data
            check2 = True
        else:
            print("Please enter right value!")
    print("THIS IS", real_or_test, "DATA")
    print("-----------------")
    for x in range(begin, end):
        new_habit[x].current_and_longest_streak()
        print("-------------")
    # checking todays work
    check1 = True
    check3 = True
    while check3 == True:
        print()
        print("which habit have you finished today? Type number,")
        habit_input = input("or n for new habit, d delete, q for quit: ")
        if habit_input == "n":
            create_habit()
        elif habit_input == "d":
            delete_habit()
        elif habit_input == "q":
            check1 = False
            check3 = False
        else:# without control, add if needend
            new_habit[int(habit_input)].checked.append(date.today())
            print("-----------------")
            new_habit[int(habit_input)].current_and_longest_streak()
            print("-----------------")
    return check1
```

This function is central switchboard for the program, it generates all branches that are needed for presenting and managing data. First part contains safe check in while loop that avoids wrong inputs and stops errors from breaking program. It is later avoided to allow program to remain decent size. Data and analytics presentation is solved by adding begin and end values that are used in following loop for analytics. Following input allows further functions (new habit adding, deleting, confirming which habit has bin done and quiting).